Student Modeling Based on Fine-Grained Programming Process Snapshots

Leinonen, Juho

2017-08-14

Leinonen , J 2017 , ' Student Modeling Based on Fine-Grained Programming Process Snapshots ' , The ICER 2017 Doctoral Consortium , Tacoma, WA , United States , 17/08/2017 - 17/08/2017 pp. 273-274 . https://doi.org/10.1145/3105726.3105732

http://hdl.handle.net/10138/316008
https://doi.org/10.1145/3105726.3105732

acceptedVersion

Student Modeling Based on Fine-Grained Programming Process Snapshots

Juho Leinonen
University of Helsinki
Helsinki, Finland
juho.leinonen@helsinki.fi

ABSTRACT

I am studying the use of fine-grained programming process data for student modeling. The initial plan is to construct different types of program state representations such as Abstract Syntax Trees (ASTs) from the data. These program state representations could be used for both automatically inferring knowledge components that the students are trying to learn as well as for modeling students' knowledge on those specific components.

KEYWORDS

student modeling; programming snapshots; programming process data; educational data mining

1 PROGRAM CONTEXT

At the University of Helsinki, we collect fine-grained data from the students' programming process [11]. On each key-press within the programming environment, information on time, assignment, student and the source code modification is stored. In previous research, we have observed that students' programming experience can be partially inferred based on this fine-grained data [6]. My plan is to investigate whether more fine-grained knowledge could be also inferred from the data, such as the mastery of specific knowledge components. Studies along this topic have previously been conducted by, for example, Rivers et al. [8], Hosseini et al. [3], and Yudelson et al. [13]

Essentially, what is remaining in my PhD is 1) to define knowledge components based on the process states either automatically or manually and then 2) use these knowledge components and information on students' success (e.g. whether the code compiles or the assignment is proceeding to a desired direction) as an input for student modeling algorithms such as Bayesian Knowledge Tracing [2, 14] and Performance Factors Analysis [7], and possibly implement novel student modeling techniques that better utilize the high granularity of the data. It is likely, however, that this will be an iterative process.

ICER'17, August 18-20, 2017, Tacoma, WA, USA.
 978-1-4503-4968-0/17/08.
DOI: http://dx.doi.org/10.1145/3105726.3105732

2 CONTEXT AND MOTIVATION

Our research group organizes the CS1 courses at the University of Helsinki and facilitates introductory programming courses throughout Finland in high schools and other universities and colleges. We also offer a MOOC with thousands of participants each year. For the time being, the courses are not too adaptive. All the students regardless of whether they study at the University or at a high-school have the same exercises, the same material, and the same support on the course. As the students' backgrounds vary a lot, especially in our MOOCs, the differences between the learners should be taken into account, and the material and the assignments should be adapted based on the students' knowledge and learning rate.

The data used in my PhD comes from these courses. In addition to the fine-grained programming process data that is being collected using a tool called Test My Code [12], we collect material usage data and voluntary demographic data such as students' gender, level of education, and age.

The current Test My Code -plugin evaluates only the functionality of the program. However, the correctness of the functionality is only a proxy for learning: it is possible that a knowledgeable student and a struggling student end up with the same solution, but took drastically different paths to achieve the solution. As we have the process data, we would like to be able to better utilize the information of the process by which the solution is constructed in addition to the end solution. This would allow us to e.g. give extra assignments to struggling students, while knowledgeable students would not need to complete assignments about a topic they already master.

3 BACKGROUND & RELATED WORK

A recent literature review that analyzed data collection in the context of computing education found that only 76 out of over 3500 articles included automatic programming process data gathering [5]. Rivers and Koedinger have developed a programming tutor that automatically generates hints based on abstract syntax trees which are built based on programming code [9]. Similarly, abstract syntax trees could be built based on students' programming snapshots and used for knowledge estimations as automatic hint generation and knowledge estimation are inherently examining the same problem – is the student struggling with a concept or not. More recently, Rivers et al. [8] studied learning curve analysis for estimating students' knowledge of KCs. They note that a lot of previous research has focused on compilation errors, and suggest that research on estimating students' knowledge based on their code (and not compilation status) should be conducted. A similar observation was made by Hosseini et al. when they studied the programming paths of students in our data [3].

Preliminary results that have been achieved in a research collaboration between University of Helsinki, University of Pittsburgh, and Carnegie Mellon University have shown that domain models can be extracted from our data and that the programming process data can be used for student modeling [13]. In my PhD, I am building on those results.

4 PROBLEM STATEMENT, RESEARCH GOALS & METHODS, AND EXPECTED CONTRIBUTIONS

In my PhD work, I hope to facilitate adaptiveness on our courses. To make the courses adaptive, we need to be able to model students' knowledge, as one of the main objectives of adaptiveness is to provide the students material and exercises that are suitable to them in regards to their current knowledge. On one hand, we want that the exercises and materials are not too hard for the student, so that they are able to learn the concepts. However, we also do not want to give out exercises or materials that the students already master as it is possibly demotivational.

While similar research has been conducted previously, research into how to model students' knowledge based on the whole programming process is still in its infancy. I have the following research objectives for my PhD thesis: 1) Construct a domain model (set of knowledge components) that describes the content the students are supposed to learn based on snapshot level data; 2) Estimate students' knowledge of course concepts based on their programming process.

To achieve these objectives, I will use the study by Yudelson et al. [13] as a starting point, and build on their work by examining other student modeling methods in addition to the Rasch model [10] and the Additive Factors Model [1]. In order to use the snapshots as input for student modeling algorithms, they need to be aggregated. Our plan is to build ASTs out of the snapshots and define KCs based on these ASTs. We could then examine sub-ASTs and see which constructs / KCs the students are struggling with by e.g. examining whether sub-ASTs compile or are correct by some other evaluation metrics.

The main contribution of my PhD is that students' learning is improved on courses where fine-grained programming snapshots are being collected. Other people in our research group are developing systems that could use the knowledge estimations for making the course contents adaptive.

5 DISSERTATION STATUS

Although I am officially a first-year PhD student, I have been working with the snapshot data already during my BSc and MSc studies on multiple publications. Additionally, I have studied relevant research which has allowed me to formulate the research objectives of my thesis. I have not yet started working on the AST construction or student modeling aspects; however, our group has an existing codebase for the purpose from previous work on CFAST [4]. I have an outline of my PhD thesis completed. My expected graduation is in late 2018 or early 2019.

6 EXPECTATIONS FROM THE DC

The data I have is very fine-grained as it contains every keystroke the students type when completing the programming assignments. I wish to discuss possible aspects that this kind of fine-grained data might have that could be used for estimating students' knowledge. For example, what parts of the programming process would be most relevant for estimating knowledge of specific knowledge components and skills. Additionally, I would appreciate ideas on what would be the best way to aggregate the data into suitable input for existing student modeling algorithms such as BKT [2, 14] or PFA [7]. For example, we want to retain the fine-grained aspects of the data, but individual programming process snapshots are hard to classify to just successes and failures as required by PFA. As I am in my first-year of PhD studies, I have plenty of time to revise and improve my plans for my PhD thesis. Furthermore, in addition to student modeling, fine-grained data could be used to infer other information about the students that could be used to facilitate learning, e.g. metacognitive strategies.

Lastly, I want to get acquainted with the ICER community as I will be attending many CSEd conferences in the next four years.

REFERENCES

[1] H. Cen, K. Koedinger, and B. Junker. Comparing two irt models for conjunctive skills. In *Intelligent tutoring systems*, pages 796–798. Springer, 2008.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[3] R. Hosseini, A. Vihavainen, and P. Brusilovsky. Exploring problem solving paths in a java programming course. 2014.

[4] D. Hovemeyer, A. Hellas, A. Petersen, and J. Spacco. Control-flow-only abstract syntax trees for analyzing students' programming progress. In *Proc. of the 2016 ACM Conference on International Computing Education Research*, pages 63–72. ACM, 2016.

[5] P. Ihantola, A. Vihavainen, A. Ahadi, M. Butler, J. Börstler, S. H. Edwards, E. Isohanni, A. Korhonen, A. Petersen, K. Rivers, M. A. Rubio, J. Sheard, B. Skupas, J. Spacco, C. Szabo, and D. Toll. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proc. of the 2015 ITiCSE on Working Group Reports*, ITICSE-WGR '15, pages 41–63, New York, NY, USA, 2015. ACM.

[6] J. Leinonen, K. Longi, A. Klami, and A. Vihavainen. Automatic inference of programming performance and experience from typing patterns. In *Proc. of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 132–137, New York, NY, USA, 2016. ACM.

[7] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[8] K. Rivers, E. Harpstead, and K. Koedinger. Learning curve analysis for programming: Which concepts do students struggle with? In *Proc. of the 2016 ACM Conference on International Computing Education Research*, pages 143–151. ACM, 2016.

[9] K. Rivers and K. R. Koedinger. Automating hint generation with solution space path construction. In *International Conference on Intelligent Tutoring Systems*, pages 329–339. Springer, 2014.

[10] W. J. van der Linden and R. K. Hambleton. *Handbook of modern item response theory*. Springer Science & Business Media, 2013.

[11] A. Vihavainen, M. Luukkainen, and P. Ihantola. Analysis of source code snapshot granularity levels. In *Proc. of the 15th Annual Conference on Information technology education*, pages 21–26. ACM, 2014.

[12] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using test my code. In *Proc. of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 117–122, New York, NY, USA, 2013. ACM.

[13] M. Yudelson, R. Hosseini, A. Vihavainen, and P. Brusilovsky. Investigating automated student modeling in a java mooc. In *Educational Data Mining 2014*, 2014.

[14] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*, pages 171–180. Springer, 2013.